FLOW CONTROL IN A RESOURCE-SHARING COMPUTER NETWORK*

by
Robert E. Kahn
William R. Crowther

I. Introduction

In this paper we discuss flow control in a resource-sharing computer network [1]. The network resources consist of a set of homogeneous or inhomogeneous computers called Hosts that are geographically distributed and are interconnected by a store-and-forward communications subnet. Each Host is connected to a store and forward switching node called an Interface Message Processor or an IMP, which is located on or nearby its premises [2]. IMPs are then interconnected by leased or private-wire synchronous circuits to form the subnet.

In such a network, the combined resources of all the Host computers are available to each Host as if the network were a single distributed computer system. A process in one Host communicates with a process in another Host by sending it a sequence of one or more messages, each of which is a bounded stream of bits, preceded by an address.

For economy of implementation, each message leaves or enters a Host via a *single* multiplexed asynchronous channel to its IMP.

A desirable interface to a communication subnet for computers (and a variety of sophisticated terminal devices) involves a collaborative procedure between the sending Host and the IMPs using feedback information from the receiver. In particular, the way of jointly controlling the flow of information from the sending Host to the receiving Host need not be constrained to synchronous schemes with one data bit being accepted every T seconds.

---

*This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract No. DAHC15-69-C-0179.

layers) of protocol are needed to escort each message from its originating process to the source IMP and similarly at the destination Host to escort each message from the IMP to its eventual destination process. The first levels of protocol typically deal with Host flow control and network interfacing. They handle allocation of buffer space, process wakeup and dispatching, network interrupts, as well as testing and error recovery features [3]. Ideally, the first levels of Host protocol should function only as a software multiplexor and not handle such tasks as process buffer allocation.

Within the IMPs, flow control is a distributed computational task requiring coordination between the source and destination IMPs as to sequencing of messages, subnet connectivity, Host responsiveness, buffer allocation, etc. Messages are normally to be delivered to the destination Host in the order they were dispatched, and without duplication. If a message cannot be delivered, for example, because no path currently exists to the destination Host or because the Host is not accepting messages, this message may be discarded by the subnet, but the source Host is to be notified of that event. The allocation of buffers at the destination IMP is required to prevent network congestion. Although the availability of mass storage in the IMPs would greatly increase the mean time to congestion, more storage alone cannot prevent its occurrence. Without mass storage, of course, the flow control must be much more diligent to prevent network congestion.

In the Host computers, flow control involves coordinating the dispatch and arrival of messages with the availability of buffer space and/or program accessibility, as well as establishing and closing process to process connections and allowing for the transfer of control as, for example, using breaks or interrupts. A partition of flow control responsibility between the IMPs and the Hosts may increase the overall reliability of network operation. In general, however, the inherent operation of the interface between the Host

structure for the interface, which suggests a truly software interface, was pointed out by Licklider [4].

One method of handling Host flow control, described by Crocker [3], has each Host initially allocate some number of buffers and notify the other Host of that amount. As successive messages are transmitted, the allocation is decremented accordingly by both Hosts. The receiving Host may then allocate additional buffers as the supply dwindles. A discussion of Fixed vs. Demand Allocation is given by Meyer [5].

Another method of handling Host flow control, suggested by Kalin [6], introduces the notion of crates for transporting the contents of a buffer. A pair of processes that wish to communicate would each have provided some number of buffers for flow control and have the other Host informed of that number. In an efficient implementation only a few buffers would typically be provided at each end and each Host might provide the same number. Let us assume the crates to be in buffers at one end initially. When a crate arrives at the other end, it may then be used to send a message in the reverse direction. Crates thus flow cyclically and the strategy insures that each transmitted crate will be accepted into a buffer at the destination. Crates may be removed from operation or transmitted to the other Host. When no more crates are available at a Host, it is not allowed to send additional messages. With a single crate in use, the communication flow alternates between the two directions. With a very large number of crates in use, the flow is essentially full-duplex.

A third method of handling flow control, suggested by Walden [7], couples the flow control more directly to actions taken by the user processes at either end. A process must either always expect to receive a message or specifically ask to receive one from the sender. In the latter case, after a buffer has been set up, a

[illegible] message which is in turn has the message to dispatch. When the SEND and RECEIVE have rendezvoused, typically at the sending Host, the message is sent.

If a Host faithfully executes any one of the above schemes and accepts messages from its IMP at a rate at least equal to the arrival rate, no congestion should occur at the destination IMP. However, if a Host should err (for example, by losing count of its current allocation), or should fail to be prompt in accepting messages from its IMP, the IMP subnet can become congested. Consequently, for reliability, a portion of the flow control mechanism must reside in the IMPs so that a malfunction in one Host does not adversely affect the service obtained by other Hosts using the net.

In this paper we examine in some detail the nature of the flow control required in the IMPs and its relation to the Host flow control and subnet performance.

2. Subnet Flow Control

It is important to distinguish between the dual objectives of introducing small transmission delays to speed the delivery of short interactive messages and obtaining high bandwidth transmission for the transfer of long files of data.

The flow control mechanism must allow for both these objectives to be simultaneously satisfied. It appears to be more the rule than the exception that a modification to the flow control ~~algorithm~~ mechanism which favors high-bandwidth traffic does so at the expense of short interactive traffic and vice-versa. As a result, the process of algorithm definition and adjustment is often a delicate matter involving at least two iterations per change.

Flow control within the subnet must ~~thus~~ provide a smooth message flow, without the introduction of undue delay, from the time of each message's entry into the net until it is delivered to the destination. With a finite number of buffers in each IMP, this task properly requires that the flow of traffic be directly regulated by the availability of buffers at the destination IMP for holding messages for its Host, and affected by (but not necessarily regulated by) the availability of buffers at the intermediate IMPs for storing messages

buffers and the latter as store-and-forward buffers. Some of these buffers may be obtained from a common pool and thus they need not be dedicated.

The ~~combination~~ CO-ORDINATION of flow with the management of reassembly buffers involves signalling strategy between the IMPs at the two ends of each connection. This aspect of flow control is discussed in Section 2.1 below. The use of store-and-forward buffers in handling the flow from IMP to IMP is discussed in Section 2.2.

2.1 Source/Destination Flow Control

An early form of subnet flow control developed for use in the ARPA Network [2] involved the concepts of "links" and "RFNMs". A link is defined to be a unidirectional logical connection between a pair of Hosts. The Host may make further use of links as an addressing mechanism that specifies a particular process to process connection. A process typically communicates with another process by sending it a sequence of messages over a given link. The two processes agree upon a link for use, in advance, by means of a suitable protocol (e.g., see [3]). The link identification is INCLUDED in the address of each message by the sending Host. Two links must be used for transmitting messages in both directions.

Only one message at a time is allowed to be present in the subnet on a link. After each message is delivered, a request for the next message (RFNM) on that link is returned to the source IMP, which passes the RFNM along to the Host. The source Host must refrain from sending the next message on that link until it has received the RFNM. The source IMP will discard any message that is sent from the Host on that link BEFORE the RFNM has been RECEIVED. While waiting for a RFNM to arrive, the Host may use any links which have no outstanding RFNMs.

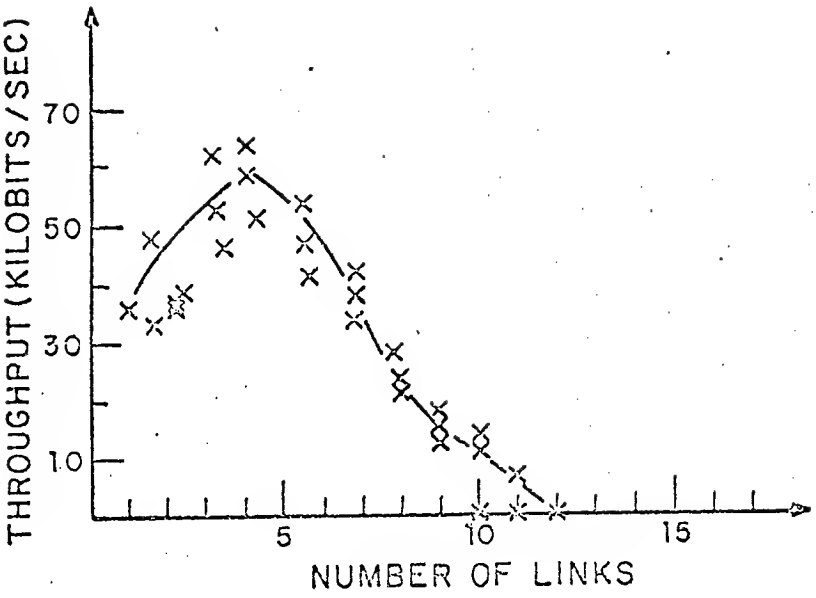
This strategy assures proper sequencing of message delivery, allows duplicates on a link to be easily detected (by use of a sequence number) and discarded, and insures that a single link cannot cause network congestion. If a source Host stops getting RFNMs, it is thus prevented by its IMP from sending additional messages on that [illegible]

The IMPs further subdivide each message into one or more packets to speed its transmission through the network. The packets of a message are independently transmitted through the subnet to the destination. The destination IMP "reassembles" the message from the individual packets and sends the message to the destination Host as a single stream of bits preceded by the addressing information. This subdivision is completely invisible to the Host computers.

This flow control technique is reliable when the destination IMP has a sufficiently large amount of reassembly storage to hold arriving messages on all the links in use. However, when a limited amount of reassembly space is available, the subnet buffer STORAGE can become FILLED ~~mixed~~ if messages are sent into the net on sufficiently many links destined for a given Host. Equivalently, the subnet buffers will become filled with backed-up messages if messages arrive for a Host (or Hosts) at a faster rate than the Host is accepting them. (P) Deadlock conditions are known to be possible in systems that involve competition for limited resources, when precautions are not taken to prevent them.

(16P) ONE type of DEADLOCK. ~~A condition~~ called *reassembly lockup* can occur in the subnet when reassembly space is unavailable to store incoming multi-packet ~~traffic~~ MESSAGES. LET US ASSUME THAT IMP A IS THE SOURCE OF MULTI-PACKET MESSAGES destined for IMP A' and that all the reassembly buffers at IMP A' are either occupied or reserved for awaited packets of partially reassembled messages. Reassembly lockup exists when the neighbors of A' are filled with store-and-forward packets also headed to A' that IMP A' cannot accept, thereby preventing packets at other IMPs from reaching the destination A' and completing the partially reassembled messages.

(IP) A simulation program that MODELS an early version of the ARPA network was used to obtain some quantitative measures of this aspect of system performance. The simulation showed the occurrence of reassembly lockup for the simple case of eight-packet traffic on many links between a pair of Hosts, as well as for traffic patterns involving multiple TRAFFIC AMONG more than two Hosts. Several values of throughput versus number of links obtained by simulation are shown in Figure 1. In this case, traffic consisting of 8 packet messages is sent from A to A'. The circuits are assumed to be 50 kilobits/sec., two paths are available for use, and in this case reassembly lockup occurs after about 10 links are in use.



21 S/F BUFFERS
32 REASSEMBLY BUFFERS
8000 BIT MESSAGES

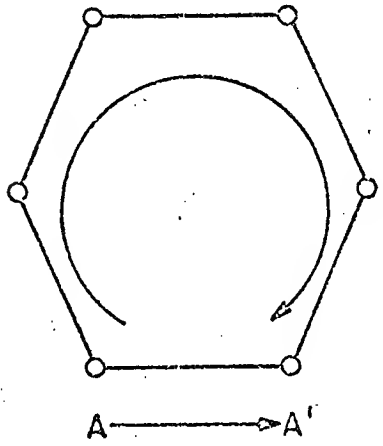


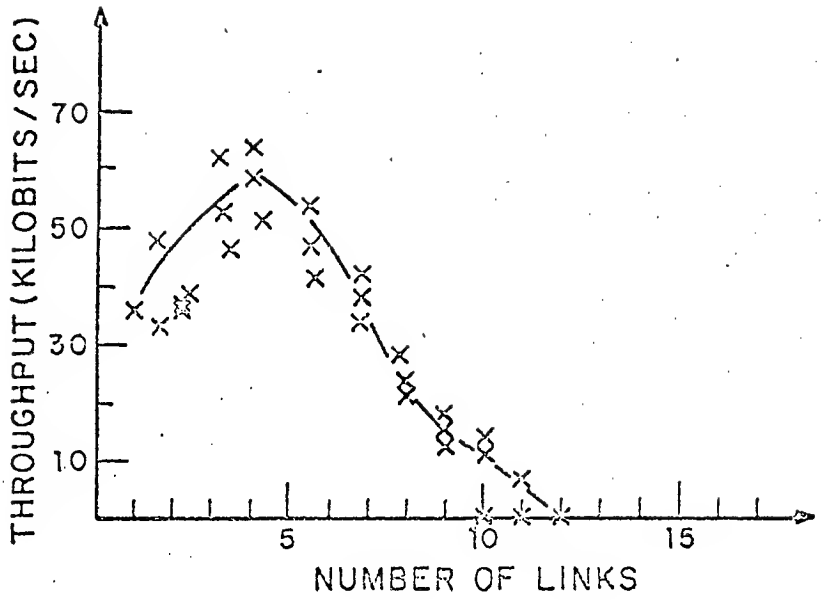
FIG.1 REASSEMBLY LOCKUP

The IMPs further subdivide each message into one or more packets to speed its transmission through the network. The packets of a message are independently transmitted through the subnet to the destination. The destination IMP "reassembles" the message from the individual packets and sends the message to the destination Host as a single stream of bits preceded by the addressing information. This subdivision is completely invisible to the Host computers.

This flow control technique is reliable when the destination IMP has a sufficiently large amount of reassembly storage to hold arriving messages on all the links in use. However, when a limited amount of reassembly space is available, the subnet buffer STORAGE can become FILLED ~~mixed~~ if messages are sent into the net on sufficiently many links destined for a given Host. Equivalently, the subnet buffers will become filled with backed-up messages if messages arrive for a Host (or Hosts) at a faster rate than the Host is accepting them. (P) Deadlock conditions are known to be possible in systems that involve competition for limited resources, when precautions are not taken to prevent them.

(16P) ~~A condition~~ One type of deadlock called *reassembly lockup* can occur in the subnet when reassembly space is unavailable to store incoming multi-packet ~~traffic~~ MESSAGES. destined for IMP A' and that all the reassembly buffers at IMP A' are LET US ASSUME THAT IMP A IS THE SOURCE OF MULTI-PACKET MESSAGES either occupied or reserved for awaited packets of partially reassembled messages. Reassembly lockup exists when the neighbors of A' are filled with store-and-forward packets also headed to A' that IMP A' cannot accept, thereby preventing packets at other IMPs from reaching the destination A' and completing the partially reassembled messages.

(1P) A simulation program that MODELS an early version of the ARPA network was used to obtain some quantitative measures of this aspect of system performance. The simulation showed the occurrence of reassembly lockup for the simple case of eight-packet traffic on many links between a pair of Hosts, as well as for traffic patterns involving multi-pt TRAFFIC AMONG more than two Hosts. Several values of throughput versus number of links obtained by simulation are shown in Figure 1. In this case, traffic consisting of 8 packet messages is sent from A to A'. The circuits are assumed to be 50 kilobits/sec., two paths are available for use, and in this case reassembly lockup occurs after about 10 links are in use.



21 S/F BUFFERS
32 REASSEMBLY BUFFERS
8000 BIT MESSAGES

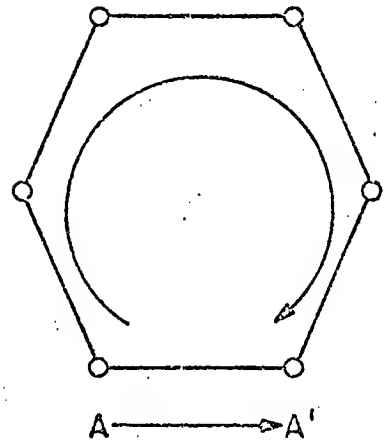


FIG.1 REASSEMBLY LOCKUP

It is characteristic of the simulation that the resulting throughput values (measured in kilobits/sec) are often quite different from one run to the next, indicating ~~the~~ a throughput statistic with sizeable variance. The length of each run was limited to be on the order of tens of thousands of messages. Only a few representative points are shown on the figure.

A solution to the reassembly lockup problem is to allow the destination IMP to discard the packets of any message which it cannot accept into reassembly and to notify the sending Host of the disposal. The sending Host can make provision to retransmit the message if it desires. The destination IMP must still perform the reassembly bookkeeping, but it need not provide the buffer storage for discarded packets. The main difficulty with this approach is that it introduces transmission failures that may be avoided with the use of other strategies.

Another solution to this problem is to have the source IMP ask the destination IMP if reassembly space is available for each message prior to sending it into the net and then to transmit the message only after buffer space has been reserved at the destination IMP. This approach introduces occasional delays in transmission rather than failures and is similar to the protocol suggested by Walden. In essence, the destination IMP executes a RECEIVE to the sending IMP only after reassembly buffer space is available. However, a space request message must first be sent from the source IMP to the destination IMP to ask for space. If reassembly space is not available, the space request is queued at the destination IMP until space becomes free.

In order to minimize delay, single packet messages serve as their own requests for space. A copy of each single packet message is held at the source IMP. If no space is available when the packet arrives at the destination, it is discarded and a message to send the copy is returned to the source IMP when space is finally reserved. This strategy insures that no setup delay is incurred whenever space is immediately available at the destination. Conversely, a setup delay is incurred only when reassembly space is unavailable. From another point of view, single packet messages

that cannot be accepted at the destination are discarded from the subnet and retransmission is provided by the source IMP.

For the first of a sequence of multi-packet messages, a short space request message is sent by the source IMP ahead of the regular message, if space has not yet been reserved, but ~~no~~ A full copy is NOT kept at the source IMP. Instead the source IMP halts the Host/IMP line until either space is known to be available, or a short duration timeout occurs. This typically introduces a setup delay on the order of tens-of-milliseconds for the *first* such multi-packet message. However, it does insure that multi-packet messages which could congest the net are not permitted to enter.

The technique of subnet flow control should allow high-bandwidth traffic between any two Hosts in an otherwise empty net. ~~But~~ However high-bandwidth cannot be obtained unless a setup delay per multi-packet message is avoided. As described above, the first multi-packet messages to a given destination will encounter a setup delay. However, each succeeding multi-packet message to that destination can avoid encountering the delay if each is sent promptly upon receipt of the RFNM. For the nonce To achieve this objective, an additional set of reassembly buffers is reserved as soon as possible after the first packet of a multi-packet message is sent to the destination Host, after which the RFNM is returned to the source IMP. The Host's next multi-packet message to that destination will be sent by the source IMP without first asking the destination IMP if buffer space exists, provided it is received by the source IMP within a short period (e.g., 125 ms.) after the Host received the RFNM. If several RFNMs for multi-packet messages have been sent to the Host, they will be timed out one at a time, in sequence. If such a timeout occurs, a message is sent to the destination IMP to free the reserved buffers. Thus, in essence, the Walden scheme is adopted for multi-packet messages with the destination IMP issuing a RECEIVE to the source IMP for each

arriving multi-packet message. A Host attempting to achieve high-bandwidth can insure that RECEIVEs will continue to be returned by promptly sending its next message. There is no possibility for a Host to achieve high-bandwidth on a given line unless it is prepared to send its next message on that line promptly.

In a subnet without mass storage, reserving space at the destination is a more powerful method for flow control than the link handling scheme (which blocks and unblocks links) since it completely frees the net from congestion and lockup due to insufficient reassembly space. As these two schemes strongly overlap in function, and the link handling scheme is costly in both program time and space, it may be deleted, with the following small attendant loss. In the link handling scheme, if a message is lost on any link due to subnet failure, that link experiences an incomplete transmission but no other links are affected. As a result of removing the link handling scheme, an incomplete transmission between two IMPs may introduce short delays on all active transmissions between the pair of IMPs. However, this event is assumed to be acceptable since the penalty is not severe and the event is expected to be quite infrequent.

The space allocation scheme must explicitly attend to sequence control and discarding of duplicate messages which it accomplishes by using sequential message numbers. If each IMP used a single message number for each other IMP and delivered all messages in the order of insertion, priority messages could occasionally experience unnecessary delays in transit, as for example, whenever one or more multi-packet messages are sent immediately preceding the priority message (and must also be delivered before the priority message). This delay points to the need to distinguish between 1) priority messages which should precede other messages into the destination Host whenever possible, and 2) short messages which must retain their original position in the sequence, as for example,

the last message of a file transfer. The Host must therefore specifically mark a priority message as such if the subnet is to be able to deliver it with priority. Of the several methods of implementing a priority handling mechanism, a simple way is for each IMP to keep two message numbers for each other IMP. The second message number is provided to handle priority messages as marked by the Host. These two message numbers correspond to two "pipes" between the IMPs and within each pipe messages are delivered in order. However, the two pipes may "slide" relative to each other so that a message in the priority pipe can be delivered before a message on the regular pipe, even though it was transmitted later in time.

The question arises as to the function of delivering RFNMs past the source IMP to the Host. Since links are no longer blocked and unblocked, the RFNM may appear to serve no useful function in designating a request for the next message, since the Host need not wait for a RFNM. Conveniently, the RFNM does still indicate that the message arrived at the destination and the Host can use links, in the old way, as always.

RFNMs are useful, however, for a Host that wishes to obtain good system response when the net is heavily loaded. For example, when a Host tries to communicate with a destination that is receiving high-bandwith traffic, it may encounter delays for messages headed to other destinations unless it dispatches messages sequenced by the return of RFNMs. If a given Host tries to obtain high bandwidth to several destinations at once, the returning RFNMs tell how to order its incoming traffic to the IMP. Furthermore, the use of RFNMs by the Host's Interfacing Software (known as the Network Control Program) will regulate a single user from causing interference with other user's communication.

When space is unavailable to queue requests for reassembly buffers at the destination IMP, a busy signal will be returned to

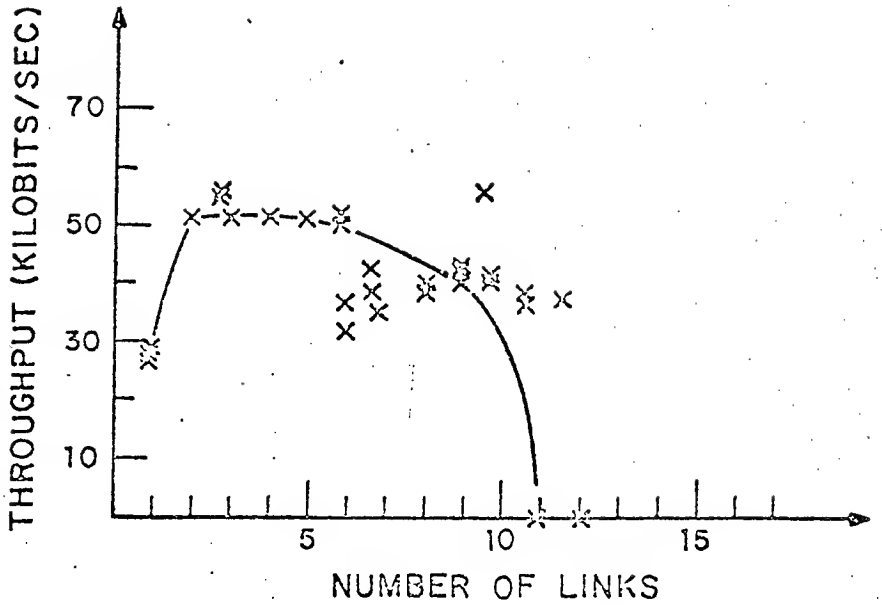
tion may be circumvented by having the source IMP continually ask for reassembly buffer space, hold onto the message and experience a short delay.

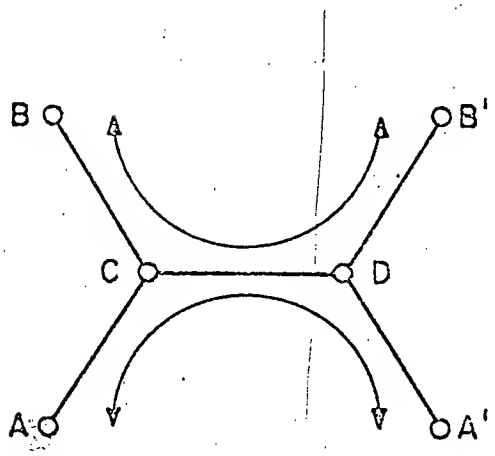
b. IMP/IMP Flow Control

We begin by describing a simple form of flow control between two adjacent IMPs. Each IMP contains N buffers, each of which may be placed on any output queue. Each IMP also keeps a copy of each transmitted packet until either an acknowledgment is returned by the neighbor in which case the copy is discarded, or a timeout occurs, in which case the packet is retransmitted. When a packet arrives without error, and a buffer is available, the packet is accepted and an acknowledgment is returned. If the packet is in error or no space is available, the packet is discarded and no acknowledgment is returned. This extremely simple technique allows maximum autonomy between the two IMPs.

Unfortunately, such a simple strategy can result in ~~conflict~~ deadlock situations which are called store-and-forward lockups. Two types of store-and-forward lockups are described below. For simplicity, we assume that ~~any~~ only single packet messages are in transit and that the destination IMP discards each arriving packet so that no subnet congestion can occur due to reassembly lockup or backup from the destination IMP.

a) Direct store-and-forward lockup — this type of lockup is illustrated by the network in Figure 2. When IMPs A and B send packets to IMPs A' and B' respectively on at least N links and vice-versa, IMP C can become filled with packets headed to IMP D and similarly IMP D can become filled with packets headed to IMP C. Due to the lack of free buffer space, neither IMP can accept packets from the other and the flow of traffic halts. Several values of throughput vs. number of ~~IMPs~~ links for 1000 bit packet traffic are included in Figure 2. The number of links is per IMP and the throughput is total throughput for all messages. Store-and-forward lockup is shown to occur with as few as eleven links per IMP each in use





21 S/F BUFFERS
∞ REASSEMBLY BUFFERS
1000 BIT MESSAGES

FIG.2 DIRECT STORE-AND-FORWARD LOCKUP

b) Indirect store-and-forward lockup — This type of lockup is illustrated by the network in Figure 3. Each IMP in the loop can become filled with packets headed to one of its neighbors and no two IMPs have packets headed to each other. A simulation of this network with the simple buffer allocation algorithm, a version of the ARPA Network routing algorithm, 21 store-and-forward buffers per IMP and 1000 bit packets shows that for 1, 4, and 16 links in operation per IMP, no lockup occurs; however, lockup occurs for 32 links. The net is obviously primed for lockup when a number of links in excess of the number store-and-forward buffers are in use. For 1, 4, and 16 links per IMP and 50 Kb/sec circuits, the total throughput was found to be 91.33, 145.68, and 44.43 kilobits/sec. For 32 links, and an initially empty net, lockup occurred after 25 messages were delivered. Depending on the details of the routing algorithm, this net could also experience a direct store-and-forward lockup.

Since no analytical procedures appear to be available for use in the selection of a buffer allocation scheme, heuristics are used to improve the efficiency of system operation. The following heuristics seem appropriate to a good system design.

1) Not all the IMP's buffers should be allowed to reside on a single output queue. Each input and output line must always be able to get some non-zero share of the buffers. One technique is to dedicate at least one buffer to each output line. Double buffering of the input lines allows all incoming messages to be examined, although not necessarily stored. Another technique, slightly more difficult to implement, is to dedicate at least one buffer for storing data to each input line. This heuristic insures that a direct store-and-forward lockup will not occur. Indirect store-and-forward lockup may be prevented by an overflow network [7].

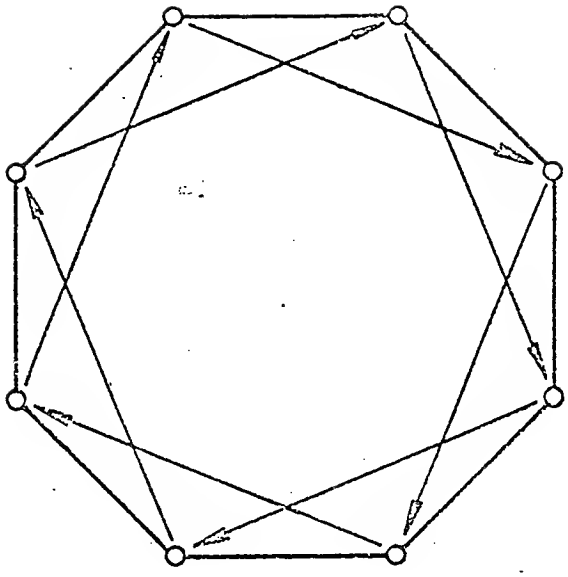


FIG.3 INDIRECT STORE-AND-FORWARD LOCKUP

b) Enough buffers should be provided so that all lines may operate at full capacity. A communication circuit requires enough store-and-forward buffering to keep the line occupied during the round-trip propagation time and to handle occasional surges and line errors. A short line probably needs no more than double buffering. For simplicity of implementation one can deploy a uniform allocation to each output line or provide a statistically sufficient pool of buffers. As the propagation time and the communication bit rate increase, the required number of buffers increases and ultimately, a form of error control that does not rely on error correction via retransmission becomes economic.

c) Negative acknowledgments are useful in activating dormant buffers more quickly, but add complexity. Although not required for system operation, they are useful in achieving a more efficient buffer utilization and are useful in routing.

d) Fifty percent of all packets in the network are acknowledgments. It is possible to save both line capacity and program utilization by using an IMP/IMP serialized transmission strategy. A serial number is attached by the IMP to each transmitted packet out a given line and the receiving IMP expects to receive packets on that line with numbers in sequence. Each arriving packet is acknowledged by the receiving IMP by returning its sequence number to the transmitting IMP. Missing sequence numbers are negatively acknowledged as are packets that are correctly received by the receiving IMP but are discarded. A similar scheme using a single phase bit (and the precise timing of a hardware rather than a software implementation) was described by Wozencraft and Horstein [9] and implemented by NPL [10]. Several

beginning of a normal message, thus reducing the amount of acknowledgment traffic by a factor of five to ten and increasing the overall capacity by upwards of 10%. Furthermore, this scheme assures that no duplicates will be generated, except when a circuit breaks.

This serialization scheme is similar to the crate scheme of Kalin except that a fixed number of buffers per line are not provided at each end. With sufficient buffering available at each IMP, a crate scheme is possible, but, in general, it makes inefficient use of IMP buffers. If no buffer space exists for a packet, it is discarded.

To insure that several lines competing for a given output are handled fairly and efficiently, a receiving IMP can designate the kind of traffic it wishes to receive from its neighbors, if any. This type of indication is similar in intent to RFNMs in a heavily loaded net, but it only passes between adjacent IMPs to allow efficient ordering of packets on the IMPs queues.

1. Roberts, L.G. and Wessler, B., Computer Network Development to Achieve Resource Sharing, Proc. SJCC, 1970.

2. Heart, F., et al, The Interface Message Processor for the ARPA Computer Network, Proc. SJCC, 1970.

3. Crocker, S., Host-to-Host Protocol Document No. 1.

4. Licklider, J.C.R.; Private Communication.

5. Meyer, E., Flow Control-Fixed versus Demand Allocation, unpublished Network Working Group Publication.

6. Kalin, R., A Simplified NCP Protocol, unpublished Network Working Group Publication.

7. Walden, D., A System for Interprocess Communication in a Resource Sharing Computer Network, unpublished Network Working Group Publication.

8. Kahn, R. and Crowther, W., A Study of the ARPA Network Design and Performance, BBN Report No. 2161, Aug. 1971.

9. Wozencraft, J. and Horstein, M.

10. D.W. DAVIES, K.A. BARTLETT, R.A. SCANTLEBURY, P.T. WILKINSON A DIGITAL COMMUNICATION NETWORK FOR COMPUTERS GIVING RAPID RESPONSE AT REMOTE TERMINALS, ACM SYMPOSIUM ON OPERATING SYSTEM PRINCIPLES 1967

Acknowledgments

The network simulation was performed with a program that models the IMP program in detail and runs slightly slower than real time on a Honeywell DDP-516 computer. The simulation program was constructed by W. Crowther and B. Cosell and the experiments were run by R. Satterfield. The simulation is basically straightforward and is not otherwise discussed in this paper.

We would like to acknowledge the participation of B. Cosell and Dr. Robert Sittler of ARCON Corporation in many helpful discussions about flow control.